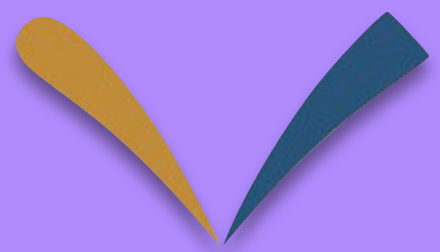


京都工芸繊維大学

Fault-prone Filtering: SPAMフィルタを用いて不具合混入モジュールを検出する試み

水野 修
京都工芸繊維大学 大学院工芸科学研究科
2009年9月9日

京都工芸繊維大学
ソフトウェア工学研究室

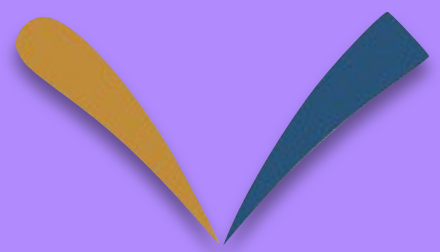
- 構成員
  - 准教授：水野 修
  - 学生：なし
    - 2009年9月にできたばかりです.
- 主なテーマ (予定)
  - フォールトプローンネス予測
    - フォールトプローンモジュールの検出手法
  - ソフトウェアプロジェクトデータの定量的解析
    - リスク要因の分析
    - 品質悪化要因(ルール)の抽出

この発表は・・・

- ESEC/FSE2007にて発表した内容を日本語に変換し，かつ，これまでの進展を加味して適宜修正したものです.

ESEC/FSE 2007

- The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2007)
  - 2年に1回はESECとの共催.
  - 2007年の採択率は17% (251本中43本の採択)
  - 場所はCavtat, Croatia
- 査読はやたらと厳しい.
  - 本論文は「条件付き」採録.
  - 担当査読委員と打ち合わせをしながら論文直せと指示が来た. 恐ろしや恐ろしや.



京都工芸繊維大学

Fault-proneness Filtering
- 概念の紹介 -

ソフトウェアの不具合の特定

- フォールトプローンモジュールの予測は，ソフトウェア工学における長年のテーマ
  - 未だにこれといった手法は無い．

- できるだけ単純な方法で特定したい．
  - 適用するときのコストを減らす．
  - 使う方からすれば，中身が何であろうが関係ない？

背景にある仮定

- 開発者は同じ間違いを犯しやすい
  - 同じ論理的間違い，同じ文法間違いなど一度犯したミスは再発する.
  - もしくは，同じミスを多くの場所で同時多発的に犯してしまう.
- 一方，一度不具合が入ったモジュールには再度不具合が入りやすい.
  - バグが混入しやすい構造がある?
- 一度犯した間違いを再度犯さないようにするためにはどうすればよいか.

我々の提案 (Fault-Proneness Filtering)

- 一度発生した不具合を学習し，類似の不具合が発生するのを防ぐ.

京都工芸繊維大学

- 精度評価
  - 一致度は 80%前後 (従来手法よりやや良)
  - 再現率 (本物の不具合を不具合と予測できる率) が高い
  - Mizuno and Kikuno, ``Training on Errors Experiment to Detect Fault-Prone Software Modules by Spam Filter,'' In The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2007), 2007.

基本的なアイディア

- SPAMメールのフィルタリング
  - 電子メール内の単語の頻度情報をBayesianフィルタで処理し，これまでに観測されたSPAM, non-SPAMグループのどちらに属するかを計算
- Fault-proneness フィルタリング
  - ソフトウェアモジュール内の単語の頻度情報をBayesianフィルタで処理し，これまでに観測されたfault-prone, non-fault-proneのどちらに属するかを計算

京都工芸繊維大学

Fault-Prone モジュールとは

- Fault-proneモジュールの定義
  - フォールトを含んでいるかもしれないソフトウェアのある一単位(モジュール)
- 本研究では
  - ソフトウェアモジュール
    - Javaのmethod
  - Fault-proneモジュール
    - 不具合を含んでいると予測したJavaのmethod
  - Faultyモジュール
    - バグトラッキングシステムなどから，ほぼ確実に不具合を含んでいると考えられるJavaのmethod

スパムE-mailフィルタリング (1)

- スパムe-mail.
  - 全世界を流れるメールの94%はスパムである.
- 多くのスパムフィルタが提案されてきた.
  - パターンマッチングによるフィルタは，スパマーとのいたちごっこに.
  - ベイジアンな手法を利用したフィルタが有効と目されている.

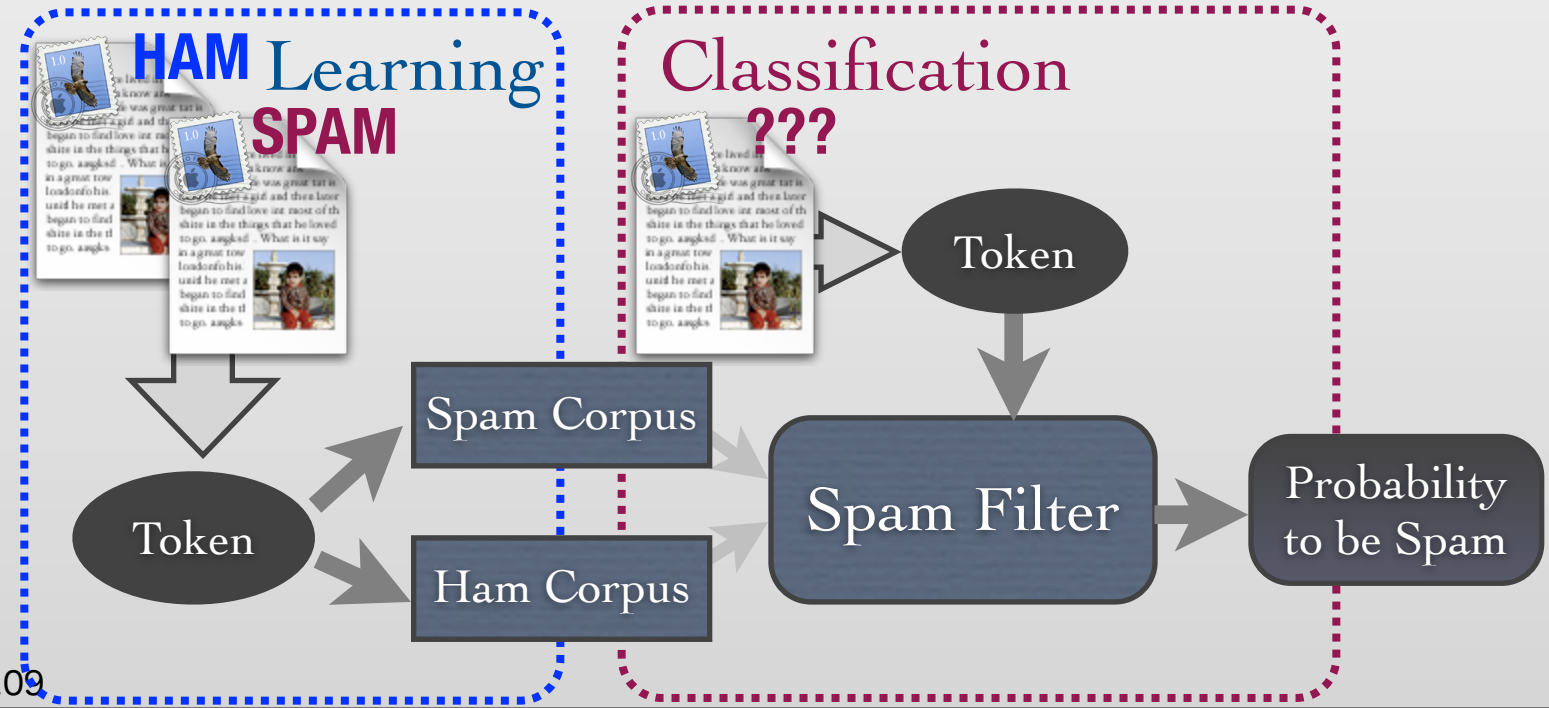
京都工芸繊維大学

- P. Graham, Hackers and Painters: Big Ideas from the Computer Age, chapter 8, pp. 121-129, 2004.

スパムE-mailフィルタリング (2)

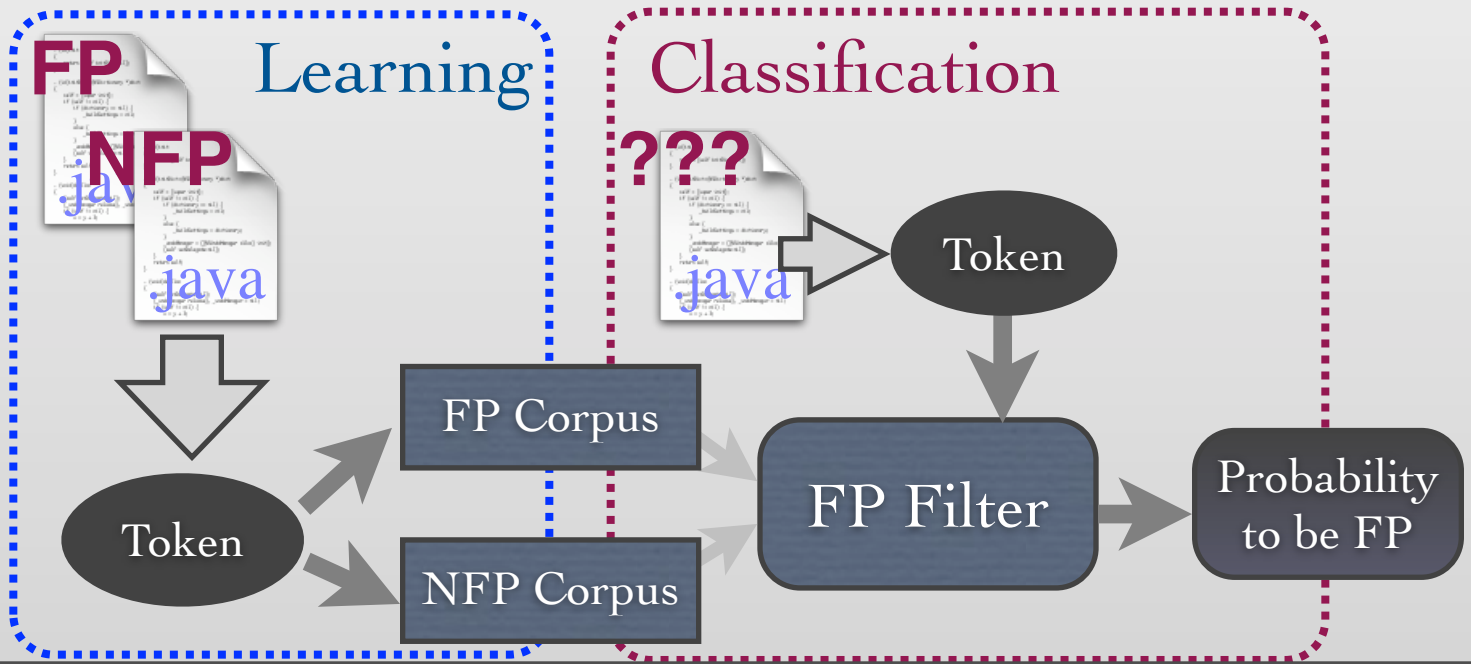
- 全てのe-mailは次の2つに分類可能：
  - Spam: 望ましくない e-mail
  - Ham: 望ましい e-mail
- Spam, Hamのそれぞれについて，単語レベルでトークン化しSpam辞書，Ham辞書に格納する．

京都工芸繊維大学



Fault-Proneness Filtering

- 概念的には，Spamフィルタと全く同様.
- ソフトウェアのソースコードモジュールを次の2つに大きく分類する.
  - バグが検出された (faulty)
  - バグが検出されていない (non-faulty)
- それぞれを単語レベルでトークン化し，faulty, non-faultyの辞書に格納する.



京都工芸繊維大学

分類の実際

- あるモジュールが入力として渡された時を考える.

FP Corpus (T_{FP})

x ++	* fact
x x	fact ++
* ++	fact x
* x	++ x

NFP Corpus (T_{NFP})

x --	* fact
x x	fact --
* --	fact x
* x	-- x

Tokens (T_{new})

x ++	+ sigma
x x	sigma ++
+ ++	sigma x
+ x	++ x

$$P(T_{FP}|T_{new}) = \frac{P(T_{new}|T_{FP})P(T_{FP})}{P(T_{new}|T_{FP})P(T_{FP}) + P(T_{new}|T_{NFP})P(T_{NFP})}$$

$$= \frac{\frac{3}{8} \times \frac{1}{2}}{\frac{3}{8} \times \frac{1}{2} + \frac{1}{8} \times \frac{1}{2}}$$

$$= 0.75$$

実験1: 予測の正確さの調査

- Eclipseプロジェクトに対する適用実験
  - 実験の目的: 統計的に正しい方法での予測精度を評価
- 実験の方法
  - Faulty, non-faultyのモジュールをあらかじめ取得
  - ランダムサンプリングによって選ばれたモジュールを使って学習
  - 同様にランダムサンプリングで選ばれたモジュールの予測

Faulty, non-faultyモジュールの収集

- Sliwerskiらのアルゴリズム(SZZ algorithm)に基づき，CVSのログとBugzillaのログから，faultyなモジュールを追跡.
  - J. Sliwerski, et. al., When do changes induce fixes? (on fridays.). In Proc. of MSR2005, pp. 24-28, 2005.
- 手順
  - 次のような単語を含むCVSログを収集して，バグが取り除かれた版を特定
    - “issue”, “problem”, “#”, とバグid(数字)
    - さらに“fixed”, “resolved”, “removed”
      - 例えば，“Issue #100 is fixed.”
  - 前の版との差分を取り，各モジュールが変更された時点を特定
  - バグidのバグが混入された時点(最初に報告された時点)より前に変更され，以降変更されていなかったモジュールにバグが存在したとする.

京都工芸繊維大学

実際の抽出

- 2006年12月時点のEclipseのBugzillaから
  - Type of faults: Bugs
  - Status of faults: Resolved, Verified, or Closed
  - Resolution of faults: Fixed
  - Severity: Blocker, Critical, Major, or Normal
  - 上記条件で，40,627個のバグを抽出
- これらのバグを含むモジュールを抽出
  - CVS log内で発見したバグ: 21,761 (全体の52%)
  - Faulty モジュール：65,782個
  - Non-faultyモジュール：1,113,063個
  - 実験にはそれぞれから20,000個前後をランダムに選択.

京都工芸繊維大学

評価

Result		予測	
		NFP	FP
実測	non-faulty	N1	N2
	faulty	N3	N4

京都工芸繊維大学

- 評価基準
  - 正答率: (N1+N4) / ΣNi
    - データの偏りなどに大きな影響を受ける
  - 再現率(recall): N4 / (N3 + N4)
    - 実不具合を予測が網羅した率
  - 適合率(precision): N4/(N2 + N4)
    - 実不具合を検出するのに要したコスト

実験1の結果

- 十重交差検証の結果
  - 高い予測精度が得られている.
    - 特に再現率(Recall)が高い

Result		予測	
		NFP	FP
実測	non-faulty	12,249	7,093
	faulty	2,972	16,243

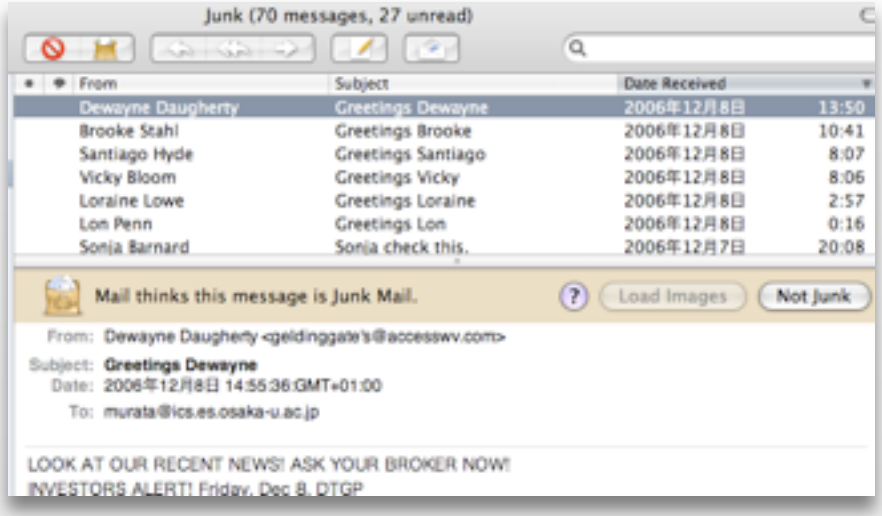
Precision: 0.696 Recall: **0.845**
Accuracy: **0.739**

- 問題:
  - 本来，バグの発生にはモジュールの生成順などが強く関与する.
  - 簡単に言えば，過去のモジュールを未来のモジュールで予測しているかもしれない.

従来手法との比較

手法	正確度	再現率	第II種の過誤率
回帰分類木(CART)	0.699	--	0.149
ロジスティック回帰	0.906	0.682	---
PCA+ロジスティック回帰	0.840	0.483	0.727
FP Filtering	0.739	0.845	0.074

実験2: Training Only Errors Procedure

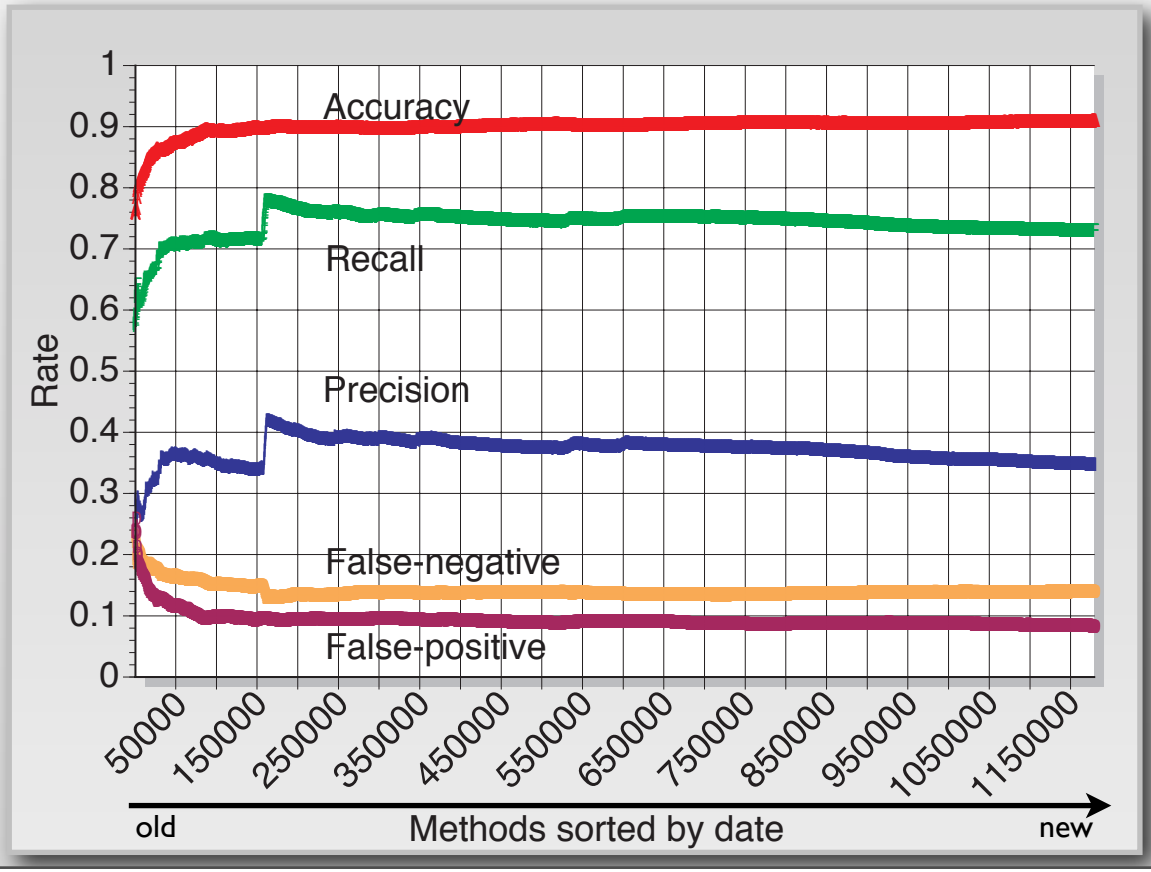


- スパムフィルタでは
  - メールは到着順にスパム判定
  - 判定を間違ったもののみを再学習
  - この手順をTraining only errorsという.
- Fault-proneness filteringでも
  - モジュールを作成順にバグ判定
  - 判定を誤ったもののみを再学習
  - この方法で，手法の評価を行う.

京都工芸繊維大学

実験2の結果(予測精度の推移)

- 全てのモジュールを時系列に並べ，初めからfaultyの判定を実施．予測が誤った時のみ再学習を実施．
- 50,000モジュール程度を判定した時点で，予測精度が安定．



京都工芸繊維大学

実験2の結果(最終的な予測精度)

- TOEの最後の時点での予測精度をまとめたもの

Result		予測 NFP	予測 FP
実測	non-faulty	1,022,895	90,168
実測	faulty	17,890	47,892

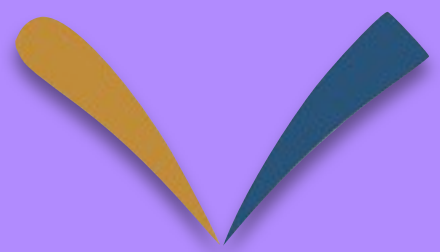
Precision: 0.347
Recall: 0.728
Accuracy: 0.908

- データの偏りが極端なため，適合率が低下
- 正答率は異様に高いが，NFP:faultyに大きく引きずられている.
- それでも，再現率の高さは確認できる.

妥当性への脅威

- オープンソースからのデータ収集にまつわる問題
  - バグは完全には追跡できない.
    - 今回も50%程度しか追跡できていない.
  - データの特性
    - ひょっとしたら，今回の手法がたまたまうまくデータにマッチしたのかもしれない.

京都工芸繊維大学



京都工芸繊維大学

Fault-proneness Filtering
- その後の展開 -

もう少し単純化した実験

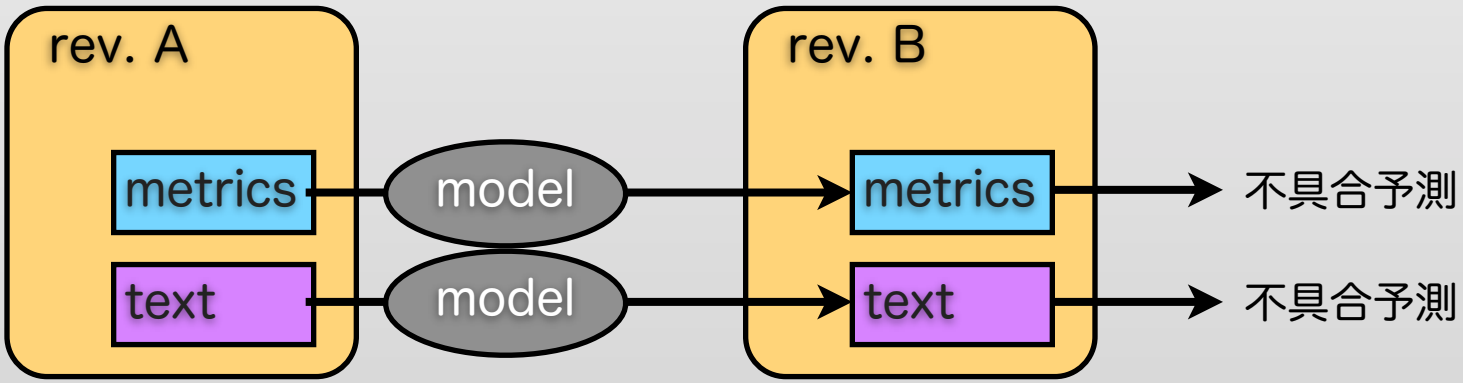
- SPAMフィルタはいろいろチューニングされているので，純粋にTextのトークンで処理しているのかが怪しい．
- 同じ条件で従来法と比較実験しよう
  - 一般的なソフトウェアメトリクス
  - テキストトークン
  - 予測手法
    - ロジスティック回帰
    - ナイーブベイズ

京都工芸繊維大学

実験方法

- プロジェクトのあるバージョンAのソースコードから，モジュールごとに「メトリクス」と「テキストトークン」を収集.
- 同様にAより未来のバージョンBにおいても収集.
- バージョンAでのデータを用いてモデルを構築.
- 上記モデルとバージョンBでのデータで不具合予測を実施.

京都工芸繊維大学



実験結果

- ほとんどのプロジェクト，モデルの組み合わせで，テキストトークンを用いた方がメトリクスを用いたものよりも高い性能を示した．

プロジェクト	モデル	データ	Recall	F値
TPTP	ロジスティック回帰	メトリクス	0.126	0.191
		テキスト	**0.658**	**0.499**
ECLP	ロジスティック回帰	メトリクス	0.089	0.142
		テキスト	**0.557**	**0.392**
BIRT	ナイーブベイズ	メトリクス	0.199	0.232
		テキスト	**0.630**	**0.299**

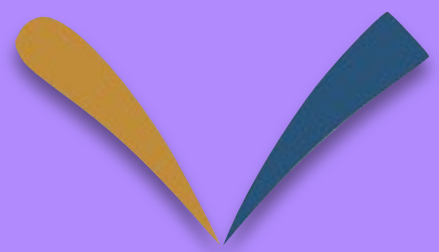
京都工芸繊維大学

副産物

- テキストトークンを用いているので，ロジスティック回帰ではどのトークンが強い影響を持っているのかが分析できる.
- 例
  - BIRTにおいては，”pointer”, “getObject”, “package”がFPに強く影響
  - BIRTにおいては，”excel”はnot FPに強く影響
- もう少し深く分析すべき

ジャーナルへの投稿

- 前述の実験結果はEmpirical Software Engineering誌に掲載が決定
  - 実はEditorがESEC/FSEのときの担当査読委員と同じ人.
  - 国際会議でおもしろいと思ってくれた人は，別のところで出会っても好意的な評価をしてくれる.
  - 国際会議でいい感触を得た論文は積極的に海外ジャーナルへの投稿をしよう.



京都工芸繊維大学

5. まとめ

Fault-proneness Filtering: まとめ (1)

- ソースコードからFPを予測する手法を提案
  - 精度は過去の研究に劣らない
  - 使用する材料はソースコードのみ

- 参考文献
  - O. Mizuno and T. Kikuno, ``Training on Errors Experiment to Detect Fault-Prone Software Modules by Spam Filter,'' In The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE2007), 2007.

Fault-proneness Filtering: まとめ(2)

- 開発環境に組み込めるようなツールを実装中.
  - CVSのラッパーや，Eclipseプラグインのようなものを想定
  - 鋭意開発中！
- 先に挙げたような課題を随時実験中.

2007年という年は…

- 水野にとって当たり年でした.
- 2006年中，子供が産まれる直前までに実験をしていて，それがうまく行ったのと同時に子供が産まれた.
- 育児サポートのために大学に出なかった時間を利用して論文がどんどん書けた．(データはすでにそろっていたので)
- 産休, Thank you!

京都工芸繊維大学

お問い合わせ

- E-mail: **o-mizuno@kit.ac.jp**
- 京都工芸繊維大学
  大学院工芸科学研究科
  情報工学部門
- 〒606-8585 京都市左京区
  松ヶ崎御所海道町
  京都工芸繊維大学 8号館 320室

京都工芸繊維大学